Adrien Protzel

Homework: Formal Relational Languages

Emp (eid: integer, ename: string, age: integer, salary: real)
Works (eid: integer, did: integer, pc_time: integer)
Dept (did: integer, dname: string, budge: real, managerid: integer)

1. Return names of every employee who works in the "Hardware", "Software", and "Research" departments.

```
    Datalog:
        Q1(ename) :- Emp(eid, ename, _, _), Works(eid, did, _), Dept(did, "Hardware",
_, _)
        Q1(ename) :- Emp(eid, ename, _, _), Works(eid, did, _), Dept(did, "Software",
_, _)
        Q1(ename) :- Emp(eid, ename, _, _), Works(eid, did, _), Dept(did, "Research",
_, _)
```

```
    Relational Algebra:
        π(ename)(σ(dname = "Hardware" V dname = "Software" V dname = "Research")(Dept)
⋈ Works ⋈ Emp)
```

```
    Relational Calculus:
        Q1(eid,ename,age,salary) = ∃eid(Emp(eid,ename,age,salary) ∧
Works(eid,did,pc_time) ∧ Dept(did,dname,budge,managerid) ∧ (dname = "Hardware" ∨
dname = "Software" ∨ dname = "Research"))
```

2. Return the names of every department without any employee.

```
    Datalog:
        Q2(dname) :- Dept(did, dname, _, _), Not Works(_, did, _)
```

```
    Relational Algebra:
        π(dname)(Dept) - π(dname)(σ(did = did)(Dept ⋈ Works))
```

```
    Relational Calculus:
        Q2(did,dname,budge,managerid) = Dept(did,dname,budge,managerid) ∧
not(∃eid(Works(eid,did,pc_time)))
```

3. Print the managerid of managers who manage only departments with budgets greater than $1.5 million.

```
    Datalog:
        Q3(managerid) :- Dept(_, _, budge, managerid), budge > 1500000
```

```
    Relational Algebra:
        π(managerid)(σ(budge > 1500000)(Dept))
```

```
    Relational Calculus:
        Q3(managerid) = ∀did(Dept(did,dname,budge,managerid) → budge > 1.5 million)
```

4. Print the name of employees whose salary is less than or equal to the salary of every employee.

```
    Datalog:
        Q4(ename) :- Emp(_, ename, _, salary), not (Emp(_, _, _, salary2), salary >
salary2).
```

```
    Relational Algebra:
        π(ename)(σ(salary <= σ(π(salary)(Emp)))(Emp))
```

```
    Relational Calculus:
        Q4(eid,ename,age,salary) = Emp(eid,ename,age,salary) ∧
∀eid2(Emp(eid2,ename2,age2,salary2) → salary ≤ salary2)
```

Notes:

```
Datalog:
Movie(mid, title, year, earned)
Actor(aid, aname, b-year)
Plays(mid, aid)

actor who played in a movie whose earned = $20
    Q1(aname) :- Actor(aid, aname, _), Plays(mid, aid), Movie(mid, _, _, 20)

actors who played in a movie whose earned = 20 AND movies made in 1998
    Q2(aname) :- Actor(aid, aname, _), Plays(mid1, aid), Movie(mid1, _, _, 20),
                                       Plays(mid2, aid), Movie(mid2, _, 1998, _)

actors who played in a movie whose earned = 20 OR movies made in 1990
    Q3(aname) :- Actor(aid, aname, _), Plays(mid, aid), Movie(mid, _, _, 20)
    Q3(aname) :- Actor(aid, aname, _), Plays(mid, aid), Movie(mid, _, 1990, _)

actors who played in a movie with earned >= 20 OR a movie made after 1990
    Q4(aname) :- Actor(aid, aname, _), Plays(mid, aid), Movie(mid, _, _, earned),
earned > 20
    Q4(aname) :- Actor(aid, aname, _), Plays(mid, aid), Movie(mid, _, year, _), year >
1990

all actors who did NOT play in a movie with "Rob"
    UQ5(aid, aname, b-year) :- Actor(aid1, aname, _), Plays(mid, aid1), Plays(mid,
aid2), Actor(aid2, "Rob", _) # make a table with all actors who did play with Rob
```

```
    Q5(aname) :- Actor(_, aname, _), NOT UQ5(aid, aname, b-year) # select all actores
not in table UQ5
```

```
Relational Algebra (RA):
Selection: σ = SELECT rows
Projection: π = SELECT columns
Cross-product: X = combines two relations in all combinations
Set-difference: - = tuples in table1 but not in table 2
Union: U = tuples in table 1 and in table 2
Join: ⋈ = cross product but on an id

Movie(mid, title, year, earned)
Actor(aid, aname, b-year)
Plays(mid, aid)

Actor:
    aid = 1, 2, 3
    aname = rob, al, jon
    b-year = 1954, 1958, 1992

Plays:
    mid = 20, 30
    aid = 1, 2

1. select actor who birth year is greater than 1990
    σ(b-year > 1990)Actor

2. select only actors names
    π(aname)Actor

3. make a single table out of Actor and Plays
    Actor X Plays

4. make a single table out of Actor and Plays, match on aid
    # R ⋈ cS = σ c(R X S)
    Actor ⋈(Actor.aid = Plays.aid)Plays
```

```
Relactional Calculus (RC):
Atom: Actor(_, _, _)
Conjunction: P ^ P
Disjunction: P V P
Implication: P → P
Negaction: not(P)
For all x P holds: ∀x.P
For an x P holds: ∃x.P = existential(not used, = _)

1. actors who played in a movie with earned = 2000
```

```
    Q1(_, _, _) = ∃_.(Actor(aid, _, _) ^ Plays(mid,aid) ^ Movie(mid, _, _, 2000))

2. actors who played only in movies produced in 1990
    Q2(x) = ∀y.Play(y, x) → ∃z.∃t.Movie(y,z,1990,t)

3. actors who played in some movies with only one actor
    Q3(x) = ∃y.Play(y, x) ^ ∀z.∀t(Play(y, z) ^ Play (y, t) → z = t)
```